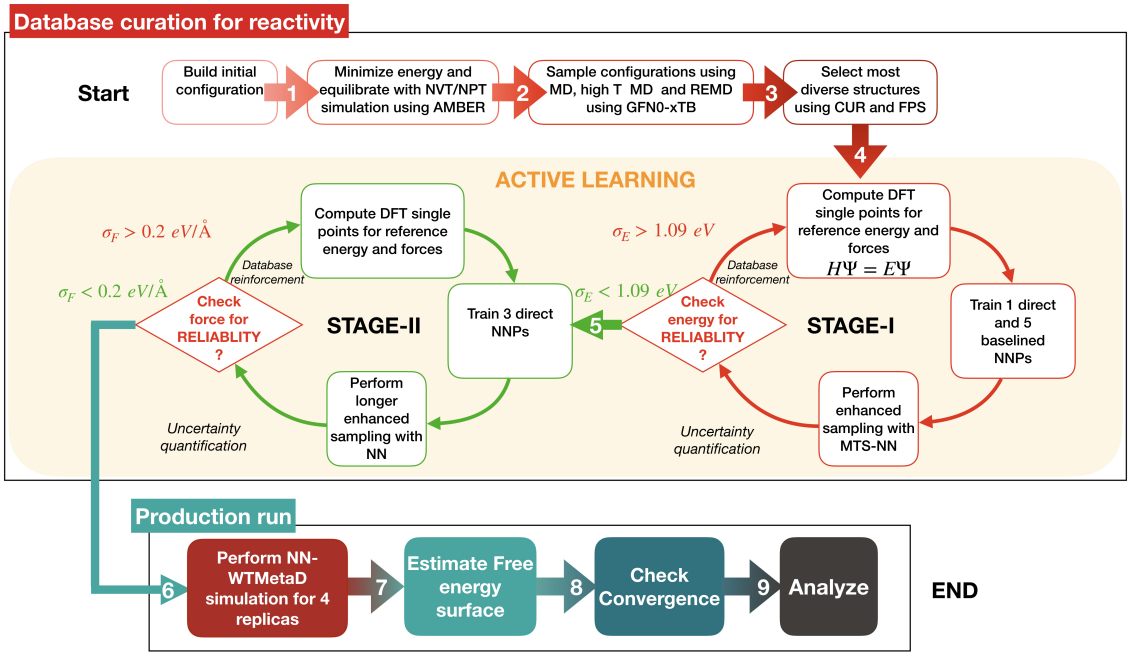
Machine Learning Potentials for solvent assisted reactions: TUTORIAL



Frédéric CELERSE
Veronika JURÁSKOVÁ
Clémence CORMINBOEUF

Contents

1 Introduction **3**

2 Softwares installation **3**

2.1 Miniconda3 . . . 3
2.2 Conda environment with python3.9 . . . 3
2.3 I–PI . . . 4
2.4 XTB . . . 4
2.5 CP2K . . . 4
2.6 DeepMD/Lammps . . . 4
2.7 Plumed . . . 4

3 Example: Modelling phthalimide's ring-opening reactivity **5**

3.1 Reaction and pipeline presentation . . . 5
3.2 Preparing the initial database . . . 6
3.2.1 Building the system . . . 6
3.2.2 XTB simulations with I–PI . . . 7
3.2.3 Selection of initial structures . . . 8
3.3 DFT reference computations . . . 9
3.4 DeePMD training . . . 10
3.4.1 Direct NN . . . 10
3.4.2 Baselined NN . . . 10
3.5 Enhanced sampling strategy: the use of Path Collective Variable with well–tempered Metadyanamics . . . 11
3.5.1 Path Collective Variables . . . 11
3.5.2 Well–tempered Metadynamics . . . 12
3.6 Active learning with MTS . . . 13
3.6.1 Compute active learning parameters . . . 13
3.6.2 Launching the simulation . . . 13
3.6.3 Selecting the structures "on–the–fly" . . . 14
3.7 Active learning with direct NNP . . . 15
3.7.1 Launching the simulation . . . 15
3.7.2 Selecting the structures "on–the–fly" . . . 15

1 Introduction

Through this tutorial we will list the different steps we need to know in order to build an efficient Machine Learning Potentials for reactive Molecular Dynamics. We will first see how to install manually all the different software which are required, and then list the different steps to curate a specific database for describing the ring-opening mechanism of a phthalimide specie.

2 Softwares installation

Here is the list of the different softwares we need to use in this tutorial:

- Miniconda3
- Python3.9
- I–PI
- XTB v6.3.2
- CP2K v9.1
- Plumed
- Lammps
- DeepMD

2.1 Miniconda3

The Miniconda3 is required to be able to build then our conda environment devoted to our environment simulation. All the informations can be found at this adress: https://docs.conda.io/projects/conda/en/latest/user-guide/install/linux.html.

2.2 Conda environment with python3.9

Once conda installed, you can create your conda environment by typing in your directory:

```
$ conda create -n deepmd python=3.9
```

The environment can be then just loaded by typing:

```
$ conda activate deepmd
```

and unloaded by typing:

```
$ conda deactivate
```

2.3 I–PI

As part of python, I–PI can be directly installed using the pip command:

```
$ pip install i-pi==2.4.0
```

2.4 XTB

The XTB software can be installed using directly the conda tool. For that, you can tap the following commands:

```
$ conda config --add channels conda-forge
$ conda install xtb=6.2.3
```

XTB issue with PBC

⚠ We need to select a version that is compatible with PBC without any bug, which is the case for 6.2.3. To ensure that this version is available, you can type in your terminal:

```
$ conda search xtb --channel conda-forge
```

We hope that future versions of XTB will solve this issue !

2.5 CP2K

The CP2K module is available on the jed SCITAS cluster by just typing:

```
$ module load gcc/11.3.0 openmpi/4.1.3
$ module load cp2k/9.1-mpi-openmp
```

2.6 DeepMD/Lammps

The DeepMD/Lammps module can be installed with the conda command. On a CPU cluster (such as jed), you can type:

```
$ conda install deepmd-kit=*=*cpu lammps-dp=*=*cpu -c deepmodeling
```

However, and especially for using deepmd in a training mode with the GPUs available on izar, we can build it on a GPU cluster by typing:

```
$ conda install deepmd-kit=*=*gpu lammps-dp=*=*gpu -c deepmodeling
```

2.7 Plumed

Plumed can be installed using the conda command. You can type:

```
$ conda install -c conda-forge plumed==2.6.2
$ conda install -c conda-forge py-plumed==2.6.2
```

3 Example: Modelling phthalimide's ring-opening reactivity

In this part we will see how to apply each step depicted on Figure 1 in order to build an efficient Machine learning Potential for describing the reactivity depicted on Figure 2.

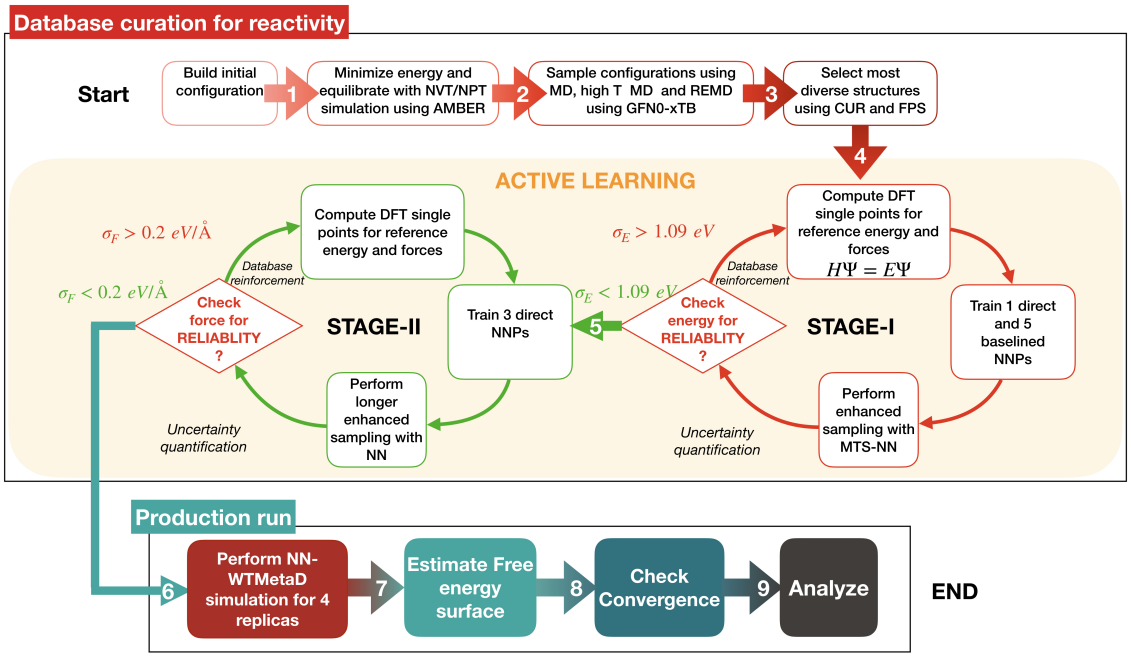


Figure 1: Pipeline for curating the corresponded database

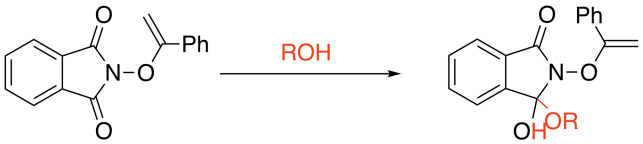


Figure 2: Reaction of interest. Only the first step is depicted because we will focus only this one. Here R=Me

3.1 Reaction and pipeline presentation

The pipeline depicted in Figure 1 can be decomposed in 4 main steps:

- STEP 1: The initial database curation by generating and sorting out–of–equilibrium structures closed to the reactant and product
- STEP 2: Configuring NN training, MD simulations and generating structures closed to the different possible TS structures.
- STEP 3: Accelerate the sampling once the database is more diverse by performing 10/20 ns of simulations (instead of only few hundred in STEP 2)

- STEP 4: Ensuring the reliability of the simulation and measuring the uncertainty

The aim of this tutorial is not to reproduce all the steps one by one, but only providing to the user the different keys to reproduce a similar work as the final database for this system already exist and can take several weeks to be generated. More information about it is available at XXX.
The reaction of interest is depicted in Figure 2. To undergo a ring–opening reaction, the first step corresponds to an addition of a solvent molecule on one of the carbonyl moiety, following by a H transfer. However, one crucial question is about the number of solvent molecules implied in this mechanism (is it only 1 molecule or is it a ring created by H bond with 2/3/4/5 molecules which is implied) ? To answer to this question, the aim is to model a full reactive molecular dynamics capturing several forward/backward events using enhanced sampling technique. The explicit treatement of the entropy will help at rationalizing the free energy barrier of the reaction, as well as measuring the statistic of the different possible TS.

3.2 Preparing the initial database

3.2.1 Building the system

WE WORK IN THE FOLDER 0_PREP_SYSTEMS The first step in curating our database relies on building an initial structure for both the reactant and the product. For each structure, an xyz structure containing the same number of atoms has to be set up. This step can be performed using any possible modules available in the field of MD. In that case, we will propose a way employing the antechamber of the AMBER software. For each system, the composition will be:

- Reactant: 1 solute + 150 solvent molecules
- Product: 1 solute + 149 solvent molecules

The number of solvent molecules was calculated following the size of the targeted box (22 $\AA$) and the density of the solvent employed here (MeOH). For the reactant, please follow the following instructions:

1. Place in a specific folder the following files: reactant.pdb, meoh.pdb, input.inp
2. Load the amber software (or at least install it):

```
$ module load amber/18/intel-17.0.8
```

3. Use packmol to generate the pdb of your system:

```
$ packmol < input.inp > out
```

4. Place the new file solvated_22A.pdb in a new folder with the files reactant.lib, reactant.frcmod, reactant.mol2, meoh.lib, meoh.frcmod, meoh.mol2 and leap.in
5. Launch the tleap module from the AMBER software:

```
$ tleap -s -f leap.in
```

6. Place the system.inpcrd, system.prmtop (just generated automatically before) with the min_cl.in file in a new folder

7. Launch the Minimization process:

```
$ sander -O -i min_cl.in -o min_cl.out -p system.prmtop -c system.
    ↪ inpcrd -r min.rst -ref system.inpcrd
```

8. Copy the min.rst in a new folder and rename it as min_cl.rst. Place also the md_nvt.inp, system.inpcrd and system.prmtop in this new folder

9. Launch the NVT small simulation (only 15/30 ps is enough):

```
$ sander.MPI -O -i md_cl.inp -o md_nvt.out -p system.prmtop -c min.
    ↪ rst -r md_cl.rst -x md_cl.nc -ref min.rst
```

10. Copy the min.rst in a new folder and rename it as min_cl.rst. Place also the md_npt.inp, system.inpcrd and system.prmtop in this new folder

11. Launch the NPT simulation (1 ns should be enough):

```
$ sander.MPI -O -i md_npt.inp -o md_npt.out -p system.prmtop -c md_cl
    ↪ .rst -r npt_cl.rst -x npt_cl.nc -ref md_cl.rst
```

12. Extract the last structure using a software which can read amber trajectory (vmd for instance, by typing vmd -f system.prmtop npt_cl.nc) as well as the final length of box. Save it with the xyz format and named it as last_reactant.xyz. You can have this information by saving in a pdb format the last frame of the simulation.

The same procedure has to be applied for the product, by just adjusting the name of files you need to use for generating at the end the last_product.xyz. **In our case, we determined for both the reactant and product a size box of 22.462 Å after NPT equilibration.**

Output Files

- last_reactant.xyz
- last_product.xyz

3.2.2 XTB simulations with I–PI

WE NOW WORK WITH THE FOLDERS LOCALIZED IN 1_XTB_SIMUS.
Once we have our initial xyz structures (e.g. last_reactant.xyz and last_product.xyz), we can generate initial out–of–equilibrium conformations of these systems by employing a semi–empirical DFT methods. The GFN1 method available in the XTB software is our choice in that case, but you can employ different other methods, even force fields, if you wish. In our case, for each system, we will generate two type of MD:

- Replica Exchange MD

- High Temperature MD

These two MD will aim at capturing structures which can prevent the future NNP to explode due to high ambient fluctuations. These two simulations are performed using the I–PI software. The required files for each simulation of each system are available in the different folders in annexe to this tutorial (e.g. 1_XTB_simus). A NOTE.txt explains how to properly launch them on a specific cluster. At the end of each simulations, you can concatenate all the structures obtained in two main files: reactant_pool.xyz and product_pool.xyz.

Output Files

- reactant_pool.xyz
- product_pool.xyz

3.2.3 Selection of initial structures

WE NOW WORK WITH THE FOLDERS LOCALIZED IN 2_FPS_SELECTION.
For each pool of structures obtained before, we would like to select the most diverse one. One suitable option is to employ the FPS scheme based on a manual input we provide. This manual input is a user–choice and should be linked to the chemical description we are targeting. In our case, we are targeting a H transfer on the O of the carbonyl as well as a C–O formation. For the reactant, we can thus measure two different distances and use them as an input for the FPS selection:

- REACTANT: the minimum distance between the O of the carbonyl and any H from the OH moiety of solvent molecules // the minimum distance between the C of the carbonyl and any O from the OH moiety of solvent molecules
- PRODUCT: the minimum distance between the H of the product alcohol and any O from the OH moiety of solvent molecules // the minimum distance between the O of the new C–O bond and any H from the OH moiety of solvent molecules

This process can be automatized using the following commands (in the case of REACTANT):

```
$ ./plumed.sh plumed_reactant.dat reactant_pool.xyz A 22.462 >
   ↪ plumed_reactant.out
$ sed -i "/PLUMED/d" plumed_reactant.out
$ python FPS.py plumed_reactant.out 100 fps_indices_reactant.dat
$ python extract_fps_selection.py fps_indices_reactant.dat reactant_pool.
   ↪ xyz reactant_fps_selected.xyz plumed_reactant.out plumed2_reactant.
   ↪ out
```

The result is depicted on Fig 3. In that case we selected only the most 100 diverse structures based on the two minimal coordination numbers. It just here corresponds to an example based on an example pool of structures for the reactant placed in the main folder. This choice does not have to be considered in a production way and based on the pool you could generate, the choice in the FPS should be more around 1000/1500 structures. Finally, the final selected structures will be placed in a file called system_selected_reactant.xyz and will be

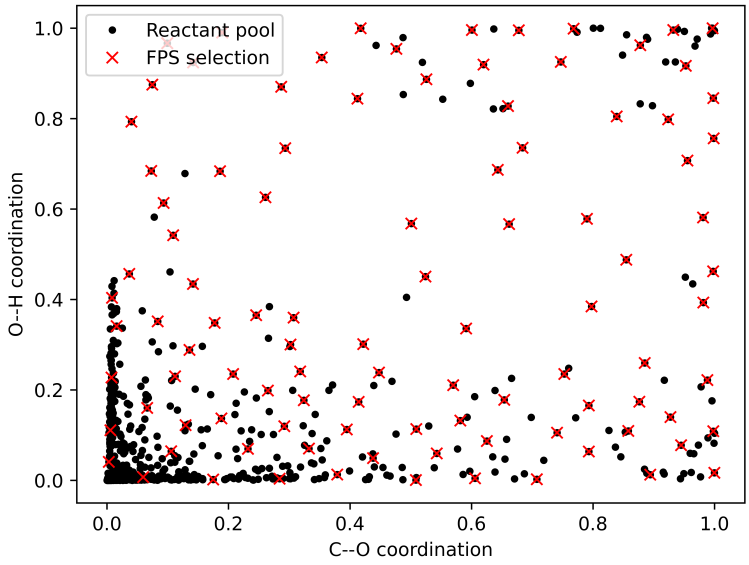


Figure 3: FPS selection

used for the next step. The same kind of file for the product (system_selected_product.xyz) is then generated following the same procedure with the corresponded files for the product.

Output Files

- system_selected_reactant.xyz
- system_selected_product.xyz

3.3 DFT reference computations

WE NOW WORK WITH THE FOLDERS LOCALIZED IN 3_DFT_COMPS.
The initial structures being selected, we have now to compute their reference energy and forces. We use the CP2K software to do this task because of it is well MPI parallelized and suitable for performing periodic DFT computations. The level of theory we target is PBE in that case, with a TZVP-MOLOPT basis set. Regarding to the intrinsic parameters owned to CP2K, they can be optimized based on the smooth tutorial available here: https://www.cp2k.org/howto:converging_cutoff

How to Converge the CUTOFF and REL_CUTOFF

⚠ This step does not have to be neglect as it is important to optimize your computational time for each single point you will perform with CP2K !

The CP2K input model is depicted in the file named pbe.cp2k and one structure example was placed in the file input.xyz. Finally, all the computations can be launched using the following scripts:

```
$ ./prep_cp2k.sh 100 reactant_fps_selected.xyz
$ ./launch_cp2k_comps.sh 100
```

At the end, all the data can be grabed together using the following scripts:

```
$ ./grep_energy.sh 100
$ python extract_dft_forces.py 100 reactant_fps_selected.xyz
```

Output Files

- reactant_fps_selected.xyz
- dft_energy.dat
- dft_forces.dat

3.4 DeePMD training

WE NOW WORK WITH THE FOLDER LOCALIZED IN 4_DEEPMD.
WARNING: THE STRUCTURES HERE DO NOT CORRESPOND TO THE OTHERS TAKEN BEFORE AND ARE JUST HERE AS AN EXAMPLE TO SHOW HOW THE SCRIPTS WORK !

We have now two types of NN to setup: direct and baselined. Let us show how to prepare the data and to launch the respective trainings.

3.4.1 Direct NN

Direct NN means that the NN will predict the full energy and forces based on an input derived from the xyz positions of each atom. To prepare the data, we go in the folder 1_DIRECT and we proceed like that:

```
$ python random_selection.py 100
$ python prepare_deepmd_for_training.py 100
$ python prepare_deepmd_for_validation.py 100
$ cd training/ && sh ../raw_to_set.sh
$ cd ../validation/ && sh ../raw_to_set.sh
$ cd ../
$ dp train training.json
```

Once the training ends, we can generate the NN file that we will need for our MD simulations. You can use the following commands:

```
$ dp freeze -o graph.pb
$ dp compress -i graph.pb -o graph-compress.pb
```

3.4.2 Baselined NN

Baselined NN means that the NN will predict only the energy difference between the target (DFT) and a baselined (in our case we will choose XTB). In that case, we need first to

compute all the XTB energies and forces. In the folder 2_BASELINED/XTB, you can do it like that:

```
$ ./prep_xtb_comps.sh 100
$ python prep_strucs.py 100
$ ./sub_xtb_comps.sh 100
$ ./grep_energy_xtb.sh 100
$ python extract_forces_xtb.py 100
```

Once you grabed all the energies and forces in the respective energy_xtb.dat and xtb_forces.dat files, you come back in the initial folder 2_BASELINED and you can generate the training and validation set with these commands:

```
$ python for_deepmd.py
$ python training_validation.py
$ cp ../1_DIRECT/training/type.raw training/
$ cp ../1_DIRECT/training/type.raw validation/
$ cd training/ && sh ../raw_to_set.sh
$ cd ../validation/ && sh ../raw_to_set.sh
$ cd ../
$ dp train training.json
```

At the end, you can generate the graph-compress.pb following the same procedure as the direct NN.

3.5 Enhanced sampling strategy: the use of Path Collective Variable with well–tempered Metadyanamics

In order to speed up the reactivity process, we use an enhanced sampling method called well–tempered metadynamics (WTMD). This method induces that we need to select one or several collective variables (CVs) that fit the best with the corresponded reactivity. In that way we defined 2 CVs based on the Path Collective Variables (PCV) approach. Let explain a bit how to deal with these two methods briefly.

3.5.1 Path Collective Variables

In few words, PCVs are a set of path-like variables that can be used to investigate complex chemical and biological processes and compute their associated free energy surfaces and kinetics. The reactivity is decomposed into several connected states, and the simulation is then biased along this defined path.
In our case, the path CV is defined using to types of coordination numbers - CN_{CO} between one carbon of the phthalimide carbonyl group and the oxygen atom from all the solvent molecules, and CN_{OH} between the oxygen atom of the same phthalimide carbonyl group as in previous CV and the hydrogen atom from the solvent OH group. The coordination number between each defined pair of atoms x and y is computed as:

$$CN(r_{xy}) = \frac{1 - \left(\frac{r_{xy} - d_0}{r_0}\right)^n}{1 - \left(\frac{r_{xy} - d_0}{r_0}\right)^m}, \tag{1}$$

d_0, r_0, n and m are fixed parameters that are chosen by the user to give the following trend to the $CN(r_{xy})$ function:

- $CN(r_{xy})$ tends to 0 when no coordination is observed
- $CN(r_{xy})$ tends to 1 when a complete coordination is observed

The values of all potential pairs in both CNs are sorted, and only the maximum value is considered in the definition of the reference structure. The parameters selected for defining both CNs are:

- CN_{CO}: $r_0 = 1.8$; $m = 6$; $n = 13$
- CN_{OH}: $r_0 = 1.5$; $m = 8$; $n = 16$

The path to the mechanism depicted in Figure 2 of the main text is defined by three reference structures which are characterized by the (CN_{CO}, CN_{OH}) pairs: (0,0), (0.5,0.5), (1,1). They enable us to define the path CVs s(d) and z(d), with d being the number of reference structures (here 3):

$$s(d) = \frac{\sum_{i=1}^{M} i e^{-\lambda \left|\left|d-d^i\right|\right|^2}}{\sum_{i=1}^{M} e^{-\lambda ||d-d^i||^2}} \tag{2}$$

$$z(d) = -\frac{1}{\lambda} log\left(\sum_{i=1}^{M} e^{-\lambda \left|\left|d-d^i\right|\right|^2}\right) \tag{3}$$

The parameter λ, which has to be properly chosen in order to well parsed each state within the path space, was set to 1.626.

3.5.2 Well–tempered Metadynamics

Metadynamics (Meta) is an enhanced sampling algorithm aiming at speeding up the sampling of complex systems and, thus, obtaining reliable and converged free energy surfaces[?] . In its original formulation, a historic–dependant bias potential V is generated along a set of initial collective variables (CVs) by gradually adding gaussians through the dynamics to accelerate the crossing of barriers:

$$V(r) = \sum_{i=1}^{n} e^{-\beta V_{i-1}(r_i)} G(r, r_i) \tag{4}$$

with $\beta = 1/k_B T$ and $G(r, r_i) = We^{-\frac{||r-r_i||^2}{2\sigma^2}}$ a Gaussian function centered on r_i, with W and σ the height and the width of the Gaussian. The free energy surface is then computed using a reweighting procedure. However, it was shown that in several cases the system can be pushed to explore high free-energy regions, leading to obtain overestimated free energy barriers. This drawback can be damped by scaling the bias using a bias factor, which will reduce the bias in the long–time sampling:

$$V(r) = \sum_{i=1}^{n} e^{-\beta/(\gamma-1) V_{i-1}(r_i)} G(r, r_i) \tag{5}$$

with $\gamma = -\frac{T + \Delta T}{T}$ the bias factor, which si still > 1. Using a tuned bias factor was shown to enhanced the free energy estimation, but it does not correct the in theory the drawback. This method is called in the literature well–tempered Metadynamics (WTMD)[?] and is implemented in Plumed 2.6.2[?] .

3.6 Active learning with MTS

3.6.1 Compute active learning parameters

Before starting the simulations, we need to compute one main active learning parameter, which is called alpha. This parameter, briefly, defined how deep the NN prediction is accurate and, thus, should be corrected. More mathematical explanations can be found in the work made by Imbalzano et al.
In the folder 5_MTS_SIMULATION/COMPUTE_ALPHA, we can find the different scripts enabling to compute the alpha value.

- First, we can create one folder for each baseline NN, and place in these folders the corresponding graph-compress.pb, predict_NN.py, from_raw_to_xyz.py, type.raw, energy.raw and coord.raw corresponding to the test set used for the baseline NN training.
- We convert the coord.raw into structures.xyz by using the from_raw_to_xyz.py script.
- We can then use the script predict_NN.py to compute the NN predictions for eac structures contained in the coord.raw file. (just adjust the num_strucs variable before launching the script).
- Once the NN predictions made in all the folders, we can compute the alpha value using the averrage_pred.py and alpha.py scripts. The alpha value will then appear in the terminal. This value will be then put in the input.xml file after the variable alpha (at line 15).
- Concerning the threshold, it is an empirical choice and, in principle, should be adjusted to the number of atoms in the system. For instance, if we target a maximum error of 2 meV/atom on each structure through the dynamics, and the system being made of 931 atoms, we obtain 931*2 = 1862 meV = 1.862 eV = 0.068 H, wich can approximated to 0.07 H. It explains our choice of 0.07 at line 17 in the file input.xml.

Repeating the procedure

⚠The procedure has to be done again each time we will retrain the baselined NNs through the active learning procedure !

3.6.2 Launching the simulation

Once the NNPs are trained, we can launch the MTS simulation. All the files needed to launch the simulation is localized in the folder 5_MTS_SIMULATION. We already described how to launch a MTS simulation implying the N2P2 NNP in our previous tutorial. Nothing in practice changes, however we will just make a small reminder about the main points to prepare and take care when we launch our simulation. A folder (here 5_MTS_SIMULATION) containing one MTS simulation should be made of these different subfolders:

1. NNPs-delta and NNPs-direct
2. plumed-1
3. lmp_delta and lmp_direct

All these folders are well connected to the main i–pi input file named input.xml. In this folder, we have to take care to well connect each driver to a specific socket and to a specific node on the cluster (as it was the case when we ran our REMD computations). Let us see how to launch our MTS simulation:

1. First, check the input.xml file and ensure that all the variables and drivers are well declared, as well as the MTS setup. Then launch the i–pi computation on the master by using the same command as we used to launch the REMD computation.
2. Then place each graph-compress.pb in the respective folder: graph-compress.pb in the NNPs-direct, graph-compress-1.pb graph-compress-2.pb graph-compress-3.pb graph-compress-4.pb in the NNPs-delta.
3. Go in lmp_direct. To create the initial.data file, we can use the get_lammps.sh file by typing sh get_lammps.sh last.xyz. This is the input file read by the lammps driver. The lmp-d1.in corresponds to the input file read by lammps and should fit with the system. Mostly, take care that the socket and the port declared at the end of the file feats well with the input.xml declared before. You can then launch the computation by using the submit.sh script.
4. Go in lmp_delta. In each of the folder, make the same as for the lmp_direct and finally launch the corresponded computation by using the submit.sh script.

Finally, you should be see on the queue 6 computations running: 1 for i–pi+XTB, 1 for direct NN and 4 for baselined NN. The different i–pi output files should start to be filled if everything was done correctly.

3.6.3 Selecting the structures "on–the–fly"

While the simulation can be viewed by using the vmd software with the simulation.position_0.xyz, the most important file in our case is the simulation.active. This file stores the structures that are extracted during the simulation, based on the active learning parameters we defined before. To obtain the corresponded xyz file, we can use the script active_to_xyz.py. Finally, the structures can be wrapped using the vmd tcl interface with the PBC tools.
Through the active learning procedure:

1. We compute for each of the selected structures the energy and forces using CP2K
2. We add these data to the already existed database
3. We reshuffle the training and test sets, and relaunch the direct and baseline NNs training procedures
4. We recompute the alpha parameter and relaunch the NN–MD, until being to observe at least one forward/backward reaction without any structures captured in the simulation.active file. The convergence of the reaction, related to the HILLS file from plumed, is not scrutinized at this point.

3.7 Active learning with direct NNP

3.7.1 Launching the simulation

Launching the simulation using only the direct NNs is much much simpler as we only used the lammps driver instead of i–pi. In that case, every files you need to have are placed in the folder 6_DIRECT_SIMULATION. The computation can be launched directly in mpi by using the submit.sh script directly.

3.7.2 Selecting the structures "on–the–fly"

During the simulation, the accuracy of each structure is measured and stored in a file called md.out. According to our procedure, we can extract manually all the structures depicting a maximum force error encompassed between 0.2 and 0.4 eV/Å by using the extract.py file. It will give you a system-extracted.lammpstrj that can be then converted in xyz using the vmd software. The procedure is the same then as for the active learning with MTS (DFT computing, NN training, ...) until being able to observe a full converged simulation with several forward/backward events.

References